University Research in Hardware Security

Ruby B. Lee
Princeton University
HotChips Hardware Security Tutorial
August 10, 2014

Outline

Hardware Security:

① Hardware-enhanced security

Protecting against attacks on software

② Secure Hardware

Protecting against attacks on hardware

We summarize different research areas in each category, and illustrate with one example in each category.

Hardware-enhanced Security Research

- **Hardware enabled isolation**
  - Static versus dynamic partitioning of resources for trusted versus untrusted software
  - Defend from “attacks from below”, e.g., untrusted OS or HV or both
- **Secure Processors (Cryptographic access control and isolation)**
  - Secure execution environment for access to keys and decrypted information
  - Hardware support for Secure key generation, management and storage
    - Master keys and key derivation
    - Secure storage
    - True Random Number Generators (TRNG)
    - Physically Unclonable Functions (PUF)
  - Mitigate information leakage thru covert channels and side channels
- **Dynamic Information Flow Tracking (DIFT)**
  - Track trusted or tainted data or control
  - Explicit versus implicit information flow tracking
  - Minimize false positives (usability) and false negatives (security)

Hardware-enhanced Security Research

- **Attestation and Trust Evidence**
  - How can the system assure the user that his desired security properties are being provided?
  - What information can a system collect to provide evidence so that the user can “trust” it?
- **Moving Target Defense**
  - Randomization and other techniques that thwart attack strategies that depend on known vulnerabilities, fixed mappings or locations, or predictable values
- **Software-hardware security verification**
  - Combine software, hardware and network protocol security verification
  - Scale to realistic systems (with accurate abstractions)
  - Compose security verification of subsystems
- **Security Metrics**
  - How can we meaningfully evaluate if one system is more secure than another, for some security property?

Secure Hardware Research

- **Detect and Mitigate Side-Channel Attacks**
  - Leak critical information through correctly implemented hardware subsystems
  - HW or SW attacks on hardware resources
    - Unlike SW vulnerabilities, the HW is functioning correctly --but leaking secrets!
  - Many types of side-channels: Power, timing, acoustic, caches, memory bus, branch prediction, E&M, fault-induced, etc.
- **Memory integrity (physical attacks)**
  - What if attacker changes the information written at a given memory address?
  - Faster Memory Integrity Trees, e.g., Bonzai Merkle tree
  - Memory integrity trees for multicore systems
- **Supply chain Security**
  - What if design is changed at some stage of chip implementation, fabrication or delivery?
  - Fake chips, old chips with limited lifetimes. Malicious chips
- **Hardware Trojans**
  - Detection and mitigation
- **Security of CAD tools** that generate and verify hardware chip designs
- **IPcores and SOC security and trustworthiness**

Cyber-Physical systems

- Security of physical systems controlled through cyberspace
  - Attacks on both software and physical components (including physical structures beyond the computer's hardware)
  - Critical infrastructures, like power-grid, transportation, water
  - Home security systems
  - Medical devices
  - Appliances in IoT (Internet of Things)
  - etc.
- Security with emerging technology in computers
  - NVRAM as main memory
  - 3D chips
  - etc.

Secure Hardware Design

Example: Secure caches
that do not leak information

Consider recent outcry over Heartbleed bug

- Heartbeat is a protocol for ensuring that a service or computer is “alive”
- However, software bug in implementing heartbeat extension in TLS/SSL in OpenSSL can be exploited to read up to 64 Kbytes of memory (per heartbeat)
  - from a client or server using a vulnerable OpenSSL version
- What is leaked?
  - **Primary key material (encryption keys, signing keys)**
  - Secondary key material (user’s name and password)
  - Protected content
  - Collateral (e.g., memory addresses, canaries, info to bypass defenses - for this session)

Remaining Vulnerabilities

But after software bug is fixed for heartbleed, the crown jewels of primary key material are still vulnerable to side-channel attacks on hardware

-- especially software side-channel attacks on hardware caches

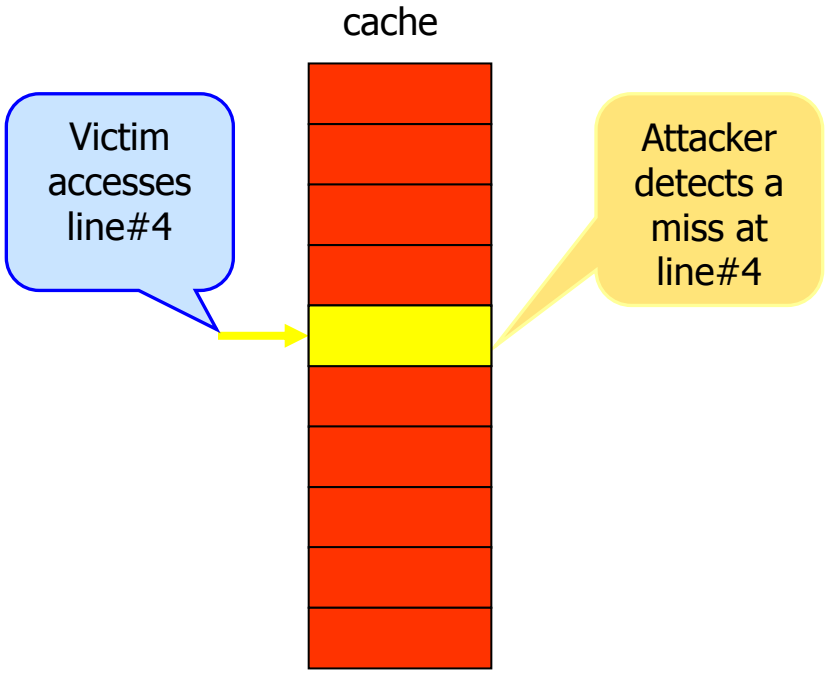
All current processors with caches are vulnerable – from embedded devices to cloud servers

The Problem -- and the Solution

- Problem:
  - Correctly functioning hardware caches leak secret information through cache side-channel attacks
    - Nullifies strong cryptography and software isolation
  - But hardware caches essential for computer performance
  - Hardware problem - very hard/slow for SW-only solutions
- Hardware Solution:
  - Secure Cache: thwart attacker, without performance hit
  - Fits in current ecosystem, works for legacy code
- Benefits:
  - Built-in; software and performance transparent

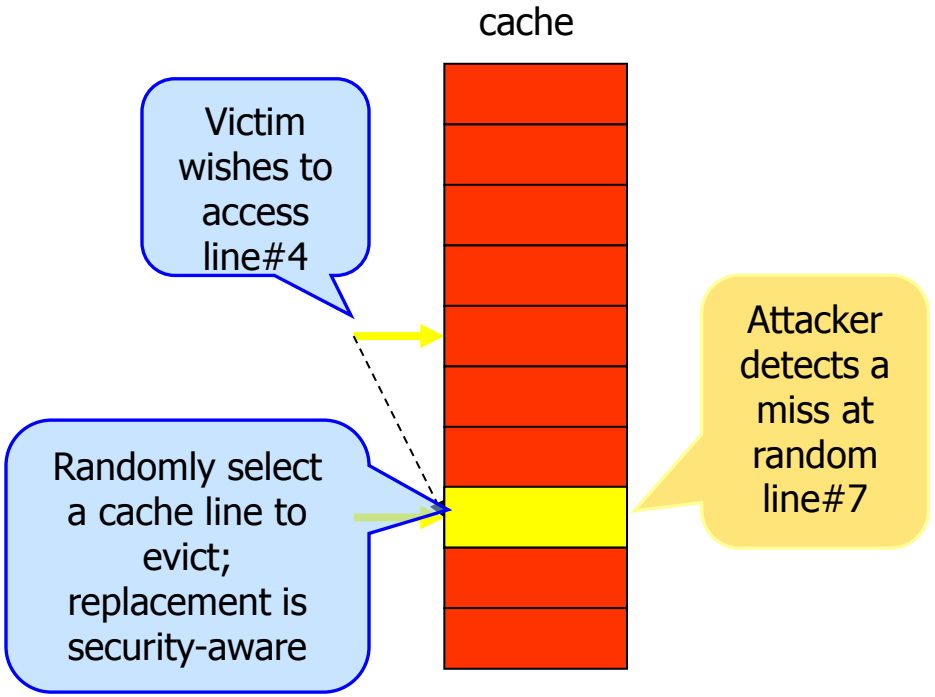
Mitigating Cache-based Attacks

- **Existing caches: fixed memory-to-cache mapping**



The attacker now knows that the victim accessed cache line #4

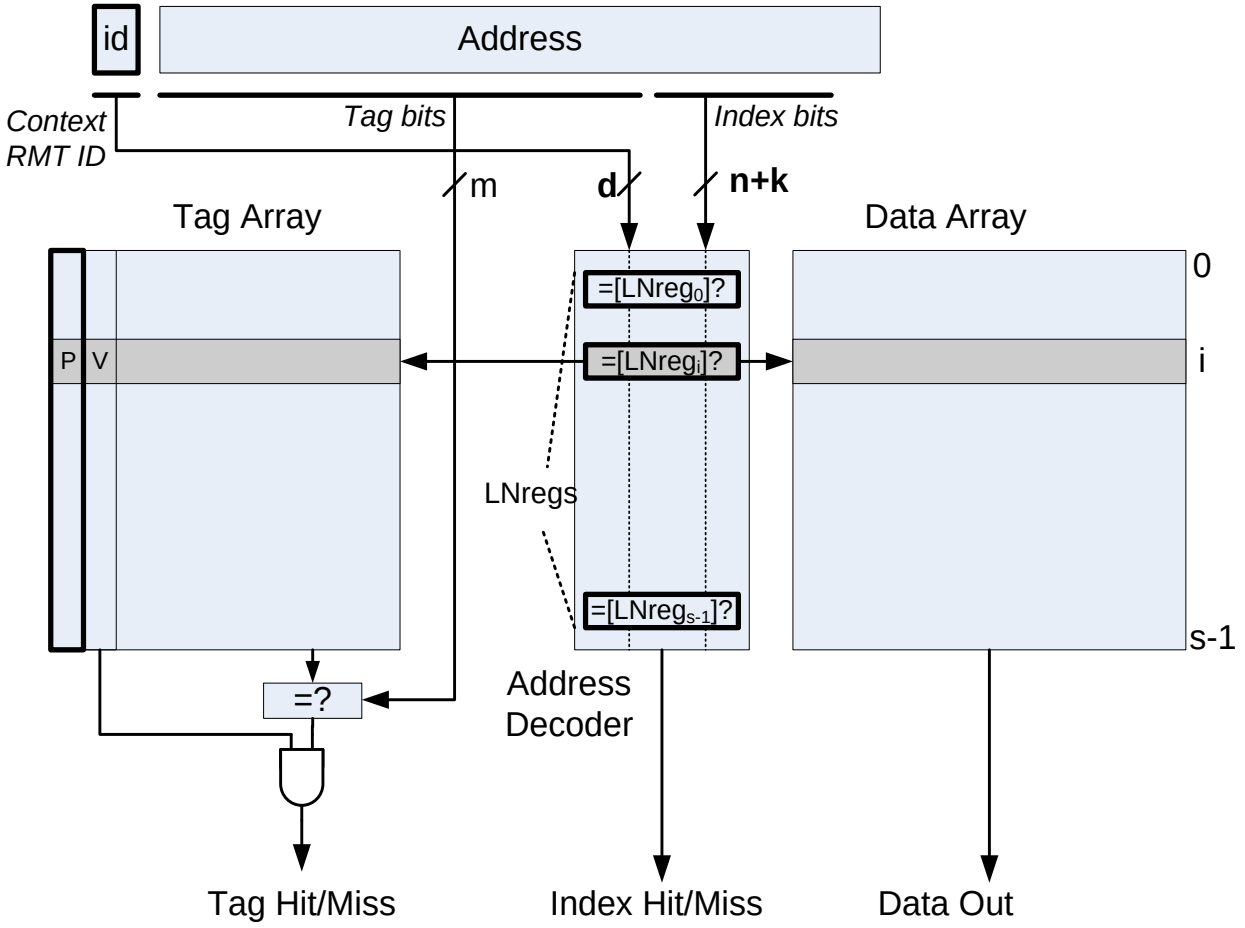
- **Newcache Solution: dynamic random mapping gives attacker no information**



By randomly selecting the line actually evicted, no information on which line is accessed by the victim can be learned by the attacker.

Newcache Architecture

- Secure cache with same performance as existing SA caches!
- novel address decoder provides dynamic, randomized memory-to-cache mapping
- longer cache index improves performance
- Holistic design: security, microarchitecture and circuit.



Newcache improves security without degrading performance or power

	Fully Associative and High Set-Assoc. Cache	Direct-Mapped Cache	Newcache
Miss rate	lowest	high	lowest
Access time	longer	shortest	short
Power per access	higher	lowest	low
Overall Power	higher	low	lowest
Security	(none)	(none)	strongest

Shortest or lowest is best

Secure cache - summary

- Example of using Moving Target Defense for Secure Hardware design (DHS/AFRL project)
  - Hardware randomizes memory-to-cache mapping
- Surprising result: need not trade off performance or power for better security
  - Contrary to conventional wisdom
  - System performance verified for smartphone and cloud server benchmarks
  - Security verified with known and new targeted attacks
  - Physical latency and power verified with test-chip
- Deployment-ready: Secure Newcache can replace existing caches

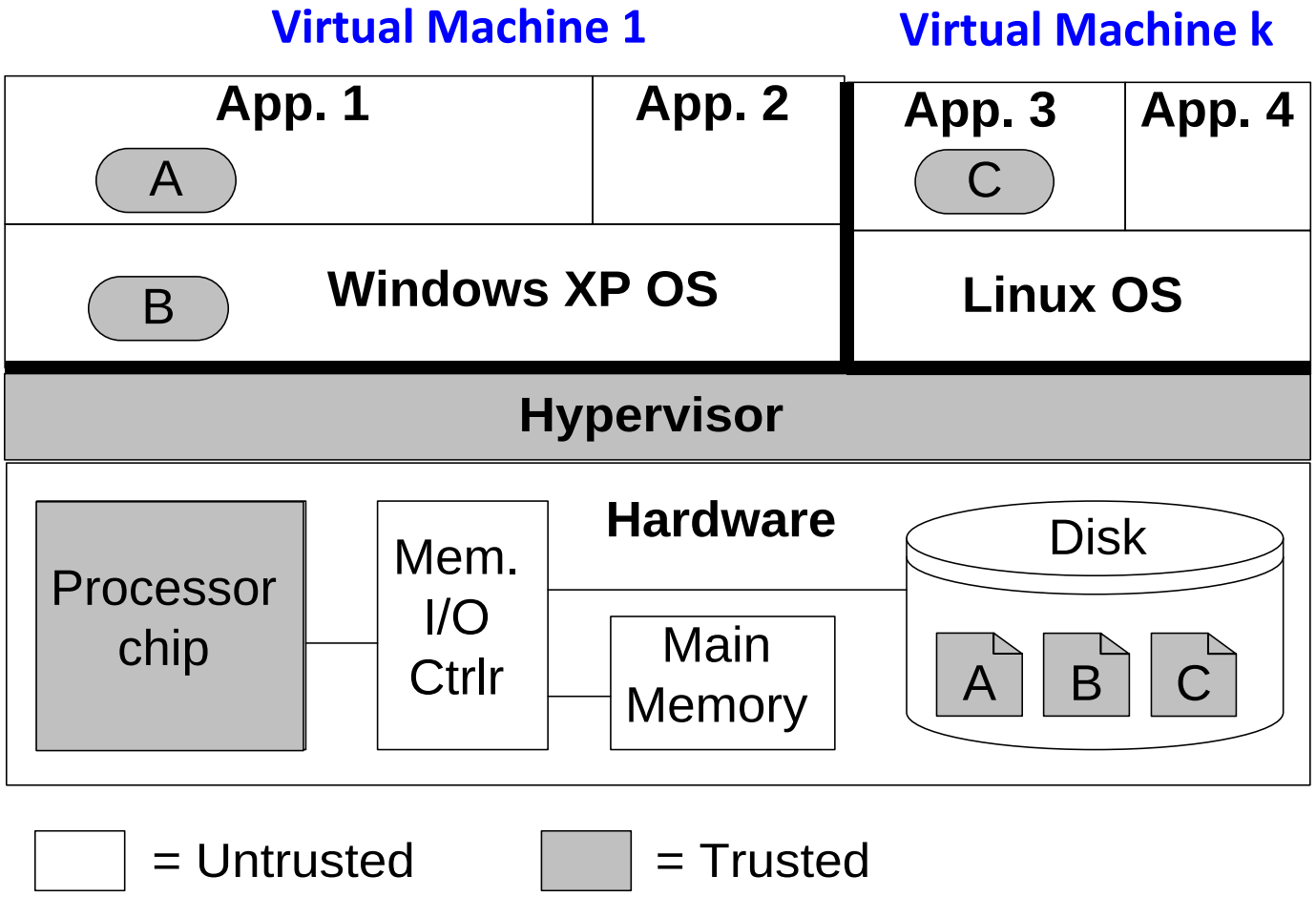
Hardware-enhanced Security Research

e.g., What is a “general-purpose” security architecture?

Bastion's Goals

- **General-Purpose HW-SW Security solution**
  - **Use software protection mechanisms (for flexibility), but use hardware to protect these.**
- Finer-Granularity Isolation, within same context
  - Protect trusted software modules within same virtual address space as untrusted app or OS
- Scalability
  - Run multiple mutually-suspicious trust domains together
- More aggressive threat model
  - O.S. as a potential adversary
  - Physical attacks in addition to software attacks
- Security when needed
  - Dynamically set up secure compartments for trusted code, rather than sandbox for untrusted code
- Resilient execution of security-critical tasks
- Provide trust evidence

Bastion security architecture

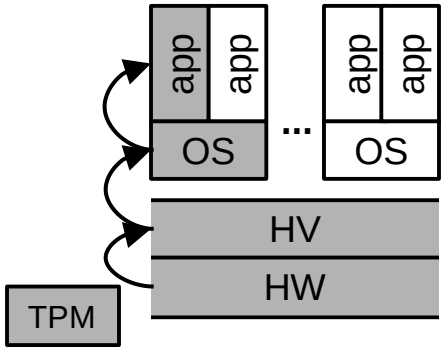


Champagne, D., Lee, R.B., "Scalable Architectural Support for Trusted Software", IEEE Intl. Symp. on High-Performance Computer Architecture (HPCA), Jan. 2010.

Hardware-enhanced Security for more aggressive Threat Models

Today

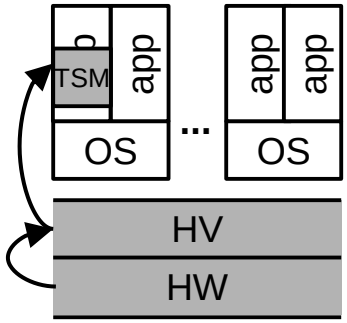
TPM,
Trustzone secure world



Trusted HV,
Trusted OS

Tomorrow? *Layer-skipping trust chains with HW trust anchors*

Fine-grained Trusted Software Modules
e.g., Bastion



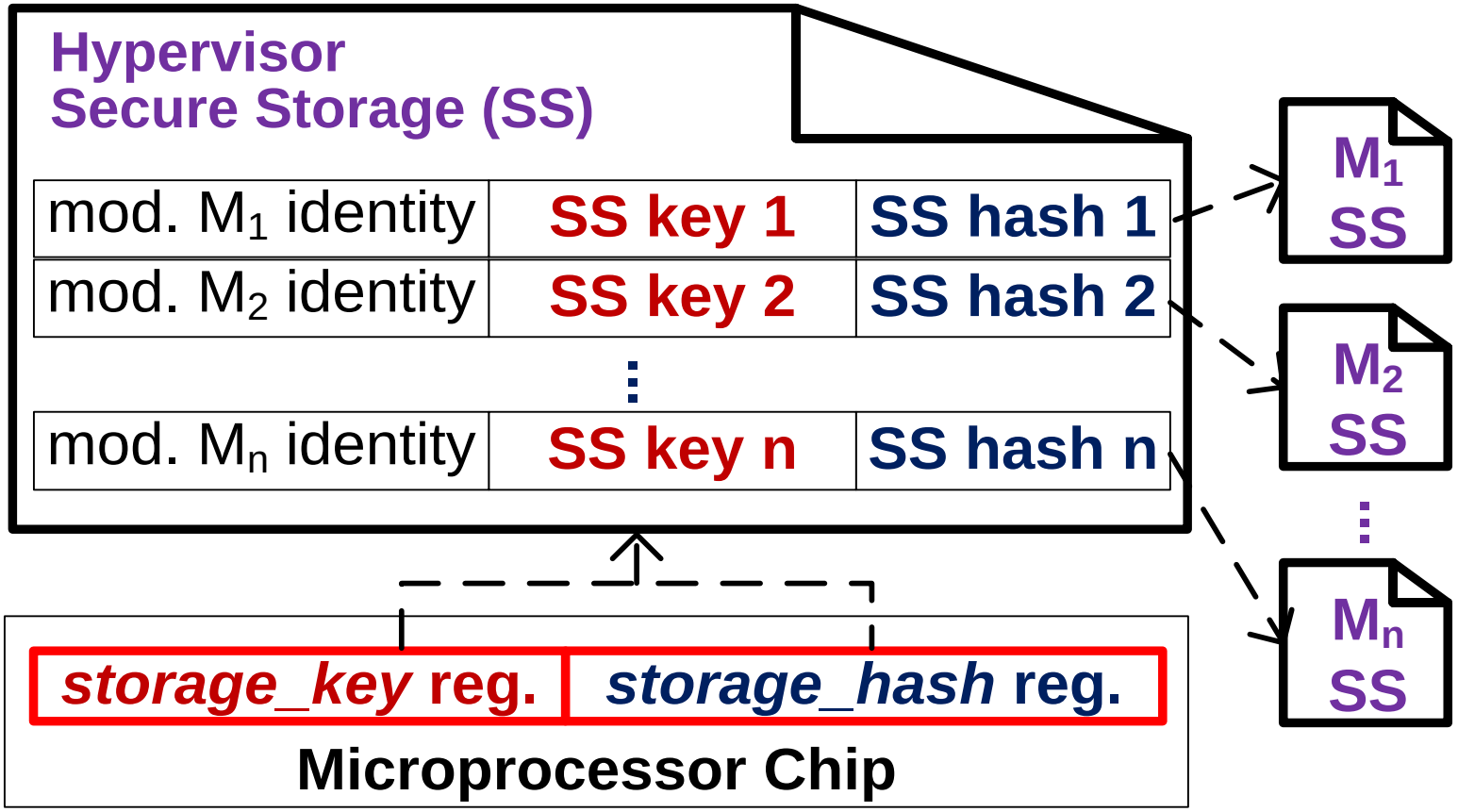
Trusted HV,
Untrusted OS

Defeats attacks from below;
More Resilient.

trusted untrusted

Scalable Secure Storage (SS)

sealed to each Trusted Software Module or Hypervisor



Trustzone: industry state-of-art

- Trustzone Advantages
  - Industry infrastructure and software ecosystem
  - Excellent for infrequent and/or self-contained security-critical tasks
    - e.g., Secure log-in; Modifying Platform configuration parameters; Establishing new Public-private key pair; BYOD (complete separation).
- But some issues:
  - One Secure World insufficient
    - If SecureOS has to be more complex, its vulnerabilities will increase
  - Performance degradation with frequent world switches
  - Loss of visibility into App or Normal OS context in Normal World
  - Security of data collected (or events triggered) by software monitor in Normal World typically cannot be trusted
  - No protection from side-channel attacks
- Enhance Trustzone by providing Secure execution environment for trusted software in Trustzone's Normal World (e.g., with Bastion).

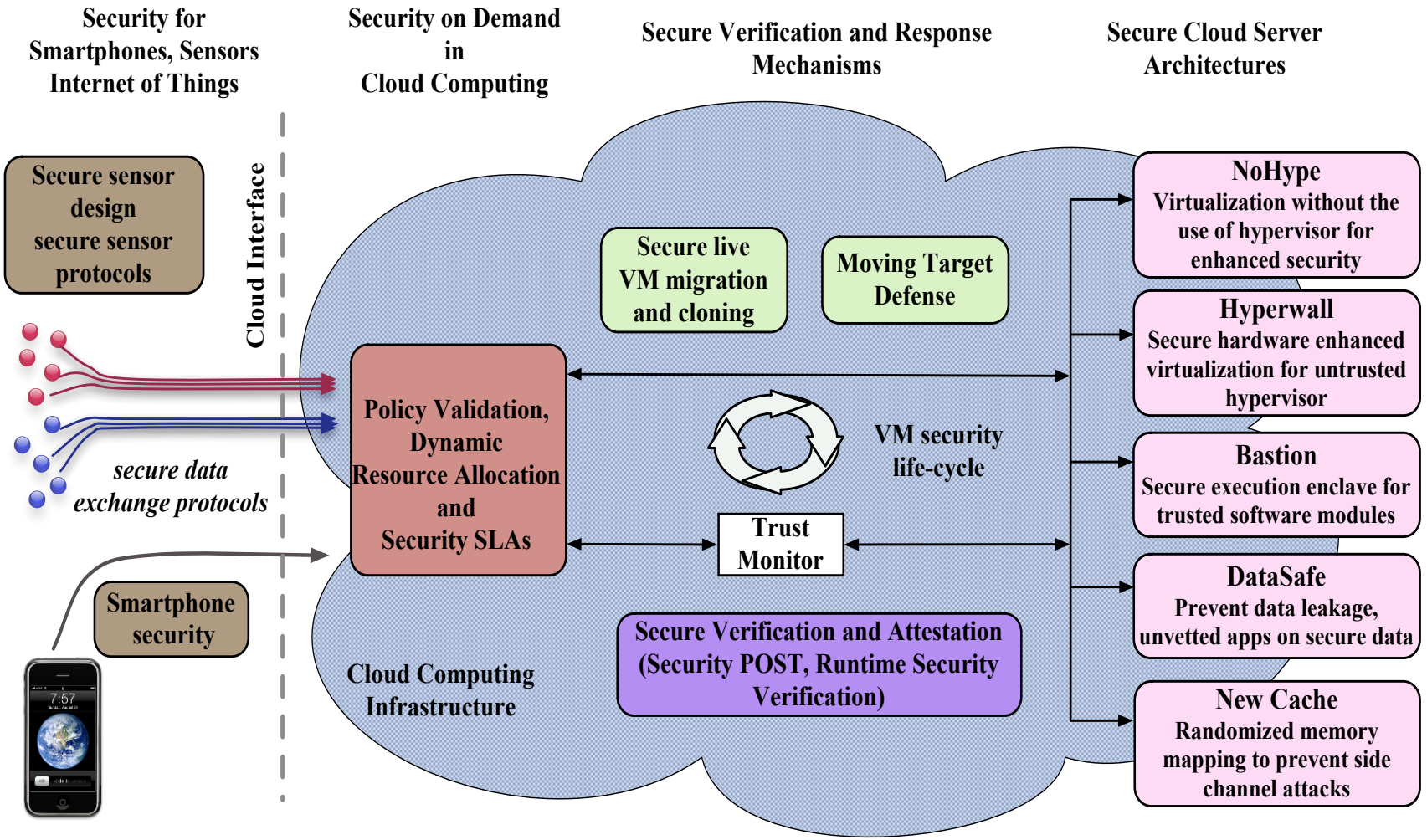
Bastion: security mechanisms

- Hypervisor Protection
  - Secure Launch of Hypervisor
  - Protecting Hypervisor at Runtime
- Trusted Software Module Protection
  - Secure Launch
  - Secure Virtual-to-Physical Memory Mapping
  - Secure Physical Memory
  - Secure Inter-Module Control Flow
- Trusted Computing Primitives
  - Secure Storage
    - sealed to each Trusted Software Module
  - Processor-based Tailored Attestation
    - Provide user with trust evidence of secure execution

Applications

- Security Monitor
  - Application-level security monitors in same address space as app.
- Protect the protection mechanisms implemented in software
  - e.g., OS based rootkit detectors
- Policy-protected Objects
  - Protected SW module can enforce arbitrary security policies for access to a protected object in secure storage
- application plug-ins
    - e-banking browser plug-in,
    - DRM media player plug-in
- Security-critical device drivers
    - e.g., HDCP for secure display hardware
- Dynamic binary translators

Hardware Security Research in Cloud Computing, Smartphones and Sensor-nets



Lee PALMS Lab, 2013.

Conclusions

- New Secure Hardware design approaches
  - e.g., Secure Newcache uses Moving Target Defense to thwart cache side-channel attacks, without degrading performance
- Design Hardware to enhance Software Security
  - e.g., Bastion: Hardware protects flexible software security monitors in same context as untrusted app being monitored
  - Can enhance Trustzone's security in its Normal World
- Many fertile security research areas in cloud computing, smartphones, sensors, IoT, multicore, SOCs, FPGAs, etc.
- Hardware security architecture should project into the future, cover different threat models, and provide proactive security.

Sample References for further reading
(all that can fit on 1 slide!)

Secure Processors

- D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural Support for Copy and Tamper Resistant Software. ASPLOS, November 2000.
- G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and Implementation of the aegis Single-Chip Secure Processor Using Physical Random Functions. ISCA, June 2005.
- R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang, "Architecture for Protecting Critical Secrets in Microprocessors," ISCA, June 2005.
- J. Dwoskin, R. B. Lee, "Hardware-rooted Trust for Secure Key Management and Transient Trust", ACM Conference on Computer and Communications Security (CCS), October 2007.
- D. Champagne, R.B. Lee, "Scalable Architectural Support for Trusted Software", HPCA, Jan. 2010.
- J. Dwoskin, D. Xu, J. Huang , M. Chiang R.B. Lee, "Secure Key Management Architecture Against Sensor-node Fabrication Attacks", Globecomm, Nov 2007.
- D. Lie, J. Mitchell, C. Thekkath and M. Horowitz. Specifying and Verifying Hardware for Tamper-Resistant Software. IEEE Symposium on Security and Privacy. May 2003.
- D. Lie, C. Thekkath, and M. Horowitz. Implementing an untrusted operating system on trusted hardware. ACM Symposium on Operating Systems Principles, Oct. 2003.
- R. B. Lee, D. K. Karig, J. P. McGregor, and Z. Shi. Enlisting hardware architecture to thwart malicious code injection. International Conference on Security in Pervasive Computing, 2003.
- S. W. Smith, E. R. Palmer, S. H. Weingart, Using a High- Performance, Programmable Secure Coprocessor. Intl. Conf. on Financial Cryptography, pp.73-89, 1998.
- S. W. Smith and S. H. Weingart,. Building a High- Performance, Programmable Secure Coprocessor. Computer Networks, 31(8), pp. 831-860, April 1999.
- D. Kirovski, M. Drinic, and M. Potkonjak, Enabling Trusted Software Integrity. ASPLOS, October 2002.
- J. D. Tygar and B. Yee. Dyad: A System for Using Physically Secure Coprocessors. Carnegie Mellon University Technical Report CMU-CS-91-140R, May 1991.

Secure Cache (Cache Side-Channel mitigation)

- Z. Wang and R.B. Lee. A Novel Cache Architecture with Enhanced Performance and Security. MICRO, 2008.
- Z. Wang and R.B. Lee. New Cache Designs for Thwarting Software Cache-based Side-Channel Attacks. ISCA, 2007.
- Z. Wang and R.B. Lee. Covert and Side Channels due to Processor Architecture. Annual Computer Security Applications Conference (ACSAC), 2006.
- L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, D. Ponomarev. Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks. ACM Transactions on Architecture and Code Optimization (TACO) Vol 8 Issue 4, Jan 2012.

Memory Integrity Tree

- Ralph C. Merkle. Protocols for public key cryptography. IEEE Symposium on Security and Privacy, 1980.
- B. Rogers, S. Chhabra, Y. Solihin, M. Prvulovic. Using Address Independent Seed Encryption and Bonsai Merkle Trees to Make Secure Processors OS- and Performance-Friendly. MICRO 2007.
- G. Suh, D. Clarke, M. van Dijk, S. Devadas, Caches and Hash Trees for Efficient Memory Integrity. HPCA, 2003.
- G. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Efficient Memory Integrity Verification and Encryption for Secure Processors. MICRO, 2003.
- Eric Hall and Charanjit S. Jutla. Parallelizable Authentication Trees . In Cryptology ePrint Archive.
- B. Rogers, C. Yan, S. Chhabra, M. Prvulovic, Y. Solihin, Single-Level Integrity and Confidentiality Protection for Distributed Shared Memory Multiprocessors, HPCA 2008.
- Elbaz, R., Champagne, D., Gebotys, C., Lee, R.B., Potlapally, N., Torres, L., Hardware Mechanisms for Memory Authentication: A Survey of Existing Techniques and Engines, *Transactions on Computational Science IV, Lecture Notes in Computer Science (LNCS)*, issue 5340, pp. 1-22, March 2009.

Crypto Acceleration in Processors

- R.B. Lee, R.B., Y. Chen. Processor Accelerator for AES. IEEE Symposium on Application Specific Processors, June 2010.
- W. Shi, H.H.S. Lee, M. Ghosh, C. Lu, A. Boldyreva. High Efficiency Counter Mode Security Architecture via Prediction and Precomputation. ISCA 2005.

Dynamic Information Flow Tracking

- M. Dalton, H. Kannan, C. Kozyrakis, Raksha: A Flexible Information Flow Architecture for Software Security. *ISCA 2007.*
- N. Zeldovich, H. Kannan, M. Dalton, and C. Kozyrakis. Hardware enforcement of application security policies using tagged memory. *OSDI*, 2008.
- M. Tiwari, H. Wassel, B. Mazloom, S. Mysore, F. Chong, and T. Sherwood, "Complete Information Flow Tracking from the Gates Up," *ASPLOS 2009.*
- N. Vachharajani, M. J. Bridges, J. Chang, R. Rangan, G. Ottoni, J. A. Blome, G. A. Reis, M. Vachharajani, , and D. I. August. RIFLE: An architectural framework for user-centric information-flow security. MICRO, 2004.
- F. Qin, C. Wang, Z. Li, H. Kim, Y. Zhou,and Y. Wu. LIFT: A low-overhead practical information flow tracking system for detecting security attacks. MICRO, 2006.
- G. E. Suh, J. Lee, and S. Devadas. Secure Program Execution via Dynamic Information Flow Tracking. ASPLOS, 2004.
- Y. Chen, P. Jamkhedkar and R. B. Lee. A Software-Hardware Architecture for Self-Protecting Data, *ACM Conference on Computer and Communications Security (CCS)*, October 2012.

Hardware Trojans

- A. Waksman, S. Sethumadhavan: Silencing Hardware Backdoors. IEEE Symposium on Security and Privacy 2011.
- A. Waksman, S. Sethumadhavan: Tamper Evident Microprocessors. IEEE Symposium on Security and Privacy, 2010

Speaker's Bio

Ruby B. Lee is the Forrest G. Hamrick Professor of Electrical Engineering at Princeton University. Her current research is in security-aware computer architecture, secure caches that do not leak information, secure cloud computing, secure virtual machines, smartphone security, running unvetted applications on sensitive data, and security verification. She has also done extensive past work on cryptographic acceleration, very fast and novel bit permutation instructions, secure processors and hardware trust anchors. Prior to Princeton, Lee served as chief architect at Hewlett-Packard for processor architecture, multimedia architecture, and then security architecture. She was a founding architect of HP's PA-RISC architecture and instrumental in the initial design of several generations of PA-RISC processors for HP's business and technical computers. She helped in the widespread adoption of multimedia in commodity products by pioneering multimedia support in microprocessors and introducing the first real-time software video in low-end products. She was co-leader of the Intel-HP multimedia architecture team for 64-bit microprocessors. She created the first security roadmap for enterprise and e-commerce security for HP. Lee is an ACM Fellow and IEEE Fellow, and holds over 120 U.S. and international patents. Known as a foremost hardware security expert, Lee is often asked to serve on national committees for improving cyber security research, such as being co-leader of the U.S. National Cyber Leap Year Summit and co-authoring the National Academies' study mandated by Congress for improving cyber security research.